Centro Universitário Newton Paiva

BACHARELADO EM SISTEMAS DE INFORMAÇÃO

BANCO DE DADOS II

Prof. Iremar Nunes de Lima

iremar.prof@uol.com.br
http://iremar.prof.sites.uol.com.br

Proibida reprodução e distribuição desta apostila por quaisquer meio.

@Copyright By Iremar Nunes de Lima

Índice

1. INTRODUÇÃO 2

1.1 CRONOGRAMA 2
1.2 BIBLIOGRAFIA BÁSICA 2
1.3 REVISÃO – MODELO CONCEITUAL E MODELO RELACIONAL 3
1.4 REVISÃO – SQL 5

2 TÓPICOS AVANÇADOS EM SQL 6

2.1 FUNÇÕES AGREGADAS 6
2.2 CONSULTAS AVANÇADAS EM SQL: PARTE I 7
2.3 CONSULTAS AVANÇADAS EM SQL: PARTE II 9
2.4 CONSULTAS AVANÇADAS EM SQL: PARTE III 10
2.5 VIEWS 12
2.6 STOREDS PROCEDURES 14
2.7 TRIGGER 16

3 TÓPICOS AVANÇADOS EM BD 18

3.1 OTIMIZAÇÃO (ÍNDICES) 18
3.2 SEGURANÇA EM BANCO DE DADOS 19
3.3 ARQUITETURA DE SGBDS 21

4. EXERCÍCIOS DE REVISÃO 23

1. Introdução

1.1 CRONOGRAMA

- Tópicos Avançados em SQL (capítulos 08, 09, 24 do livro texto).
  - ✓ Funções Agregadas.
  - ✓ Consultas avançadas.
  - ✓ Subconsultas.
  - ✓ Views.
  - ✓ Storeds Procedures.
  - ✓ Gatilhos (Triggers).
- Tópicos Avançados em BD (capítulos 02, 14, 17, 18, 19, 23 do livro texto).
  - ✓ Índices.
  - ✓ Segurança (Grants e Schemes)
  - ✓ Arquitetura de SGBD.
  - ✓ Administração de SGBD.
- Práticas e Estudos de Casos no SGBD Microsoft SQL Server.
- Artigos Complementares.

1.2 BIBLIOGRAFIA BÁSICA

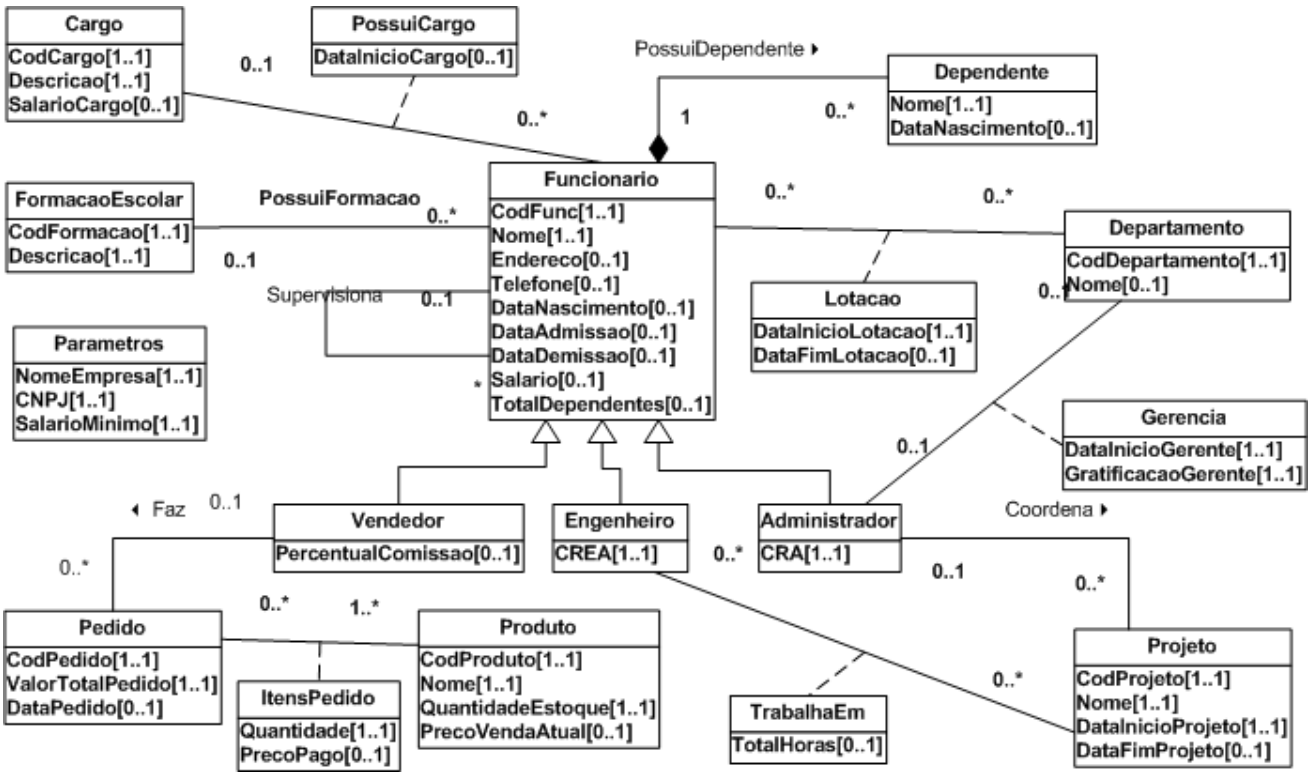
ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados.** 4. ed. São Paulo: Pearson Addison Wesley, 2005. 724 p.

KORTH, HENRY F.; SILBERSCHATZ, ABRAHAM; SUDARSHAN, S. **Sistema de Banco de Dados.** 5 ed. Elsevier, 2006. 808 p.

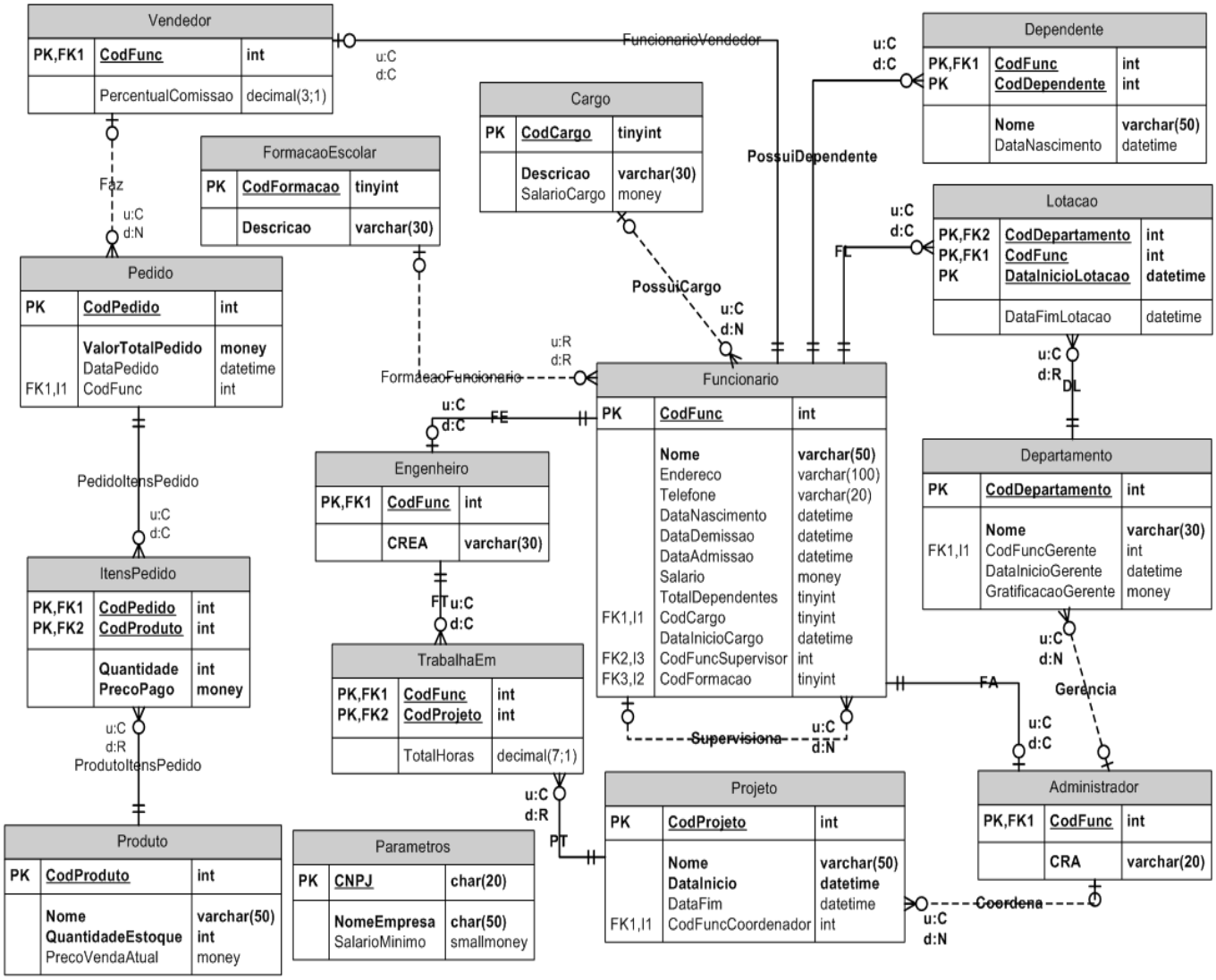
TAYLOR, Allen G. **SQL para Dummies.** Rio de Janeiro: Campus, 2001. 409 p.

1.3 REVISÃO – MODELO CONCEITUAL E MODELO RELACIONAL

Etapa 1: Construir o modelo conceitual do banco de dados (DCP segundo a notação UML):



Etapa 2: Construir o modelo relacional (DER segundo a notação Peter Chen - Pés de Galinha):



1.4 REVISÃO – SQL

Exemplo 1: Recupere o Nome e Endereço dos Funcionários que tenham salário igual ou inferior a 500.00 reais e que tenham sido admitidos após o ano de 2000.

```
SELECT    Nome, Endereco
FROM      Funcionario
WHERE     Salario <= 500.00 AND  DataAdmissao >= '01/01/2001'
```

Exemplo 2: Selecione o nome dos departamentos com o nome dos funcionários atualmente lotados nele mostrando o salário de cada funcionário.

```
SELECT    D.Nome, F.Nome, F.Salario, F.Salario
FROM      Departamento D INNER JOIN Lotacao L ON
          (D.CodDepartamento=L.CodDepartamento ) INNER JOIN
          Funcionario F ON (L.CodFunc = F.CodFunc)
WHERE     L.DataFimLotacao IS NULL
```

Exemplo 3: (Left Outer Join) Recupere o código, data e valor total dos pedidos e o nome do vendedor que fez o pedido. Mostre todos os pedidos mesmo que não tenha vendedor:

```
SELECT    P.CodPedido, P.DataPedido, P.ValorTotalPedido, F.Nome
FROM      Pedido P LEFT OUTER JOIN Funcionario F
          ON (P.CodFunc = F.CodFunc)
```

Exercícios:

1) Selecione o nome dos funcionários e o nome do departamento em que ele trabalha atualmente.
2) Selecione o nome dos projetos não encerrados e o nome dos funcionários que trabalham no projeto. Mostre todos os projetos. Ordene os dados pelo nome do projeto seguido pelo nome do funcinário.
3) Selecione o nome dos projetos e o nome dos respectivos coordenadores dos projetos. Mostre todos os projetos.
4) Recupere o nome de todos os funcionários, o nome da sua formação escolar, o nome do seu cargo, o nome dos departamentos em que está lotado atualmente e o nome dos projetos em que trabalha. Ordene o resultado pelo nome do funcionario, seguido pela formacao, seguido pelo cargo, seguido pelo departamento, e por último seguido pelo projeto.
5) Recupere os dados de todos os pedidos deste ano.
6) Recupere o nome do funcionário e o nome dos seus dependentes com mais de 10 anos de idade. Ordene os dados pelo nome do funcionário seguido pela data de nascimento do dependente.
7) Selecione o nome dos Produtos com estoque abaixo de 20 unidades.
8) Selecione o Codigo do pedido, o valor total do pedido, a data do pedido, o nome do vendedor que tirou o pedido. Mostre somente os pedidos do ano de 2006.

2 Tópicos Avançados em SQL

2.1 FUNÇÕES AGREGADAS

SQL possui 4 funções chamadas funções agregadas (COUNT, SUM, AVG, MAX e MIN).

Exemplo 1: Recupere a soma dos salários de todos os empregados, o maior salário, o menor salário, e a média dos salários:

```
SELECT    COUNT(CodFunc)     AS "Total de funcionários",
          SUM(Salario)       AS "Soma dos salários",
          MAX(Salario)       AS "Maior Salário",
          MIN(Salario)       AS "Menor Salário",
          AVG(Salario)       AS "Media dos Salários"
FROM Funcionario
```

Total de funcionários	Soma dos salários	Maior Salário	Menor Salário	Media dos Salários
17	35426.8900	6670.0000	310.0000	2214.1806

Exemplo 2: Para cada cargo recupere o código do cargo, a quantidade de empregados que possui o cargo e média salarial dos empregados que exercem este cargo:

```
SELECT    CodCargo AS "Código do Cargo", COUNT(CodFunc) AS "Total Empregados no
          Cargo", AVG(Salario) AS "Média Salarial"
FROM      Funcionario
GROUP BY CodCargo
```

Exemplo 3: Recupere o nome dos funcionários admitidos a mais de três anos e que trabalham em mais de dois projetos. Mostre o total de projetos que eles trabalham.

```
SELECT    F.Nome, COUNT(T.CodProjeto) AS 'Total Projetos'
FROM      Funcionario F INNER JOIN TrabalhaEm T ON (F.CodFunc = T.CodFunc)
WHERE     (Getdate() - F.DataAdmissao) > 3 * 365
GROUP BY F.Nome
HAVING    COUNT(T.CodProjeto) > 2
```

Exemplo 4: Recupere o Código dos departamentos e a quantidade de empregados lotados atualmente mostrando somente os departamentos que tenham mais de 3 funcionários lotados ordenados pelo total de empregados em ordem decrescente.

```
SELECT CodDepartamento, Count(CodFunc) AS "Total de empregados lotados atualmente"
FROM   Lotacao
WHERE DataFimLotacao IS NULL
GROUP BY CodDepartamento
HAVING Count(CodFunc) > 3
ORDER BY Count(CodFunc) DESC
```

Obs: 1.Na cláusula Group By é obrigatório que apareça todos os atributos da cláusula SELECT.
2.Se a cláusula Order By for utilizada ela deve aparecer depois da cláusula Group By.

Exemplo 5: Recupere o nome dos funcionários admitidos a mais de três anos e que trabalham em no máximo dois projetos.

```
SELECT    F.Nome
FROM      Funcionario F LEFT OUTER JOIN TrabalhaEm T ON (F.CodFunc = T.CodFunc)
WHERE     (Getdate() - F.DataAdmissao) > 3 * 365
GROUP BY F.CodFunc, F.Nome
HAVING    COUNT(T.CodFunc) <= 2
```

Exercícios:

1) Calcule a média salarial dos funcionários não demitidos.
2) Calcule o total de funcionários vendedores com salário acima de 1000 reais.
3) Qual o valor do maior e menor salário dentre os funcionários coordenadores de projeto?
4) Mostre o nome do departamento e o total de funcionários não demitidos lotados atualmente no departamento.
5)Recupere o nome de todos os funcionários, o total de projetos em que trabalharam e o total de horas que já trabalharam nos projetos.
6) Recupere o nome dos departamentos que contenham até cinco funcionários lotados atualmente.
7) Recupere o nome dos projetos com mais de 2 funcionários não demitidos trabalhando nele.
8) Recupere quantos são e a media salarial dos funcionários que são vendedores e coordenadores de projeto ao mesmo tempo.

2.2 CONSULTAS AVANÇADAS EM SQL: PARTE I

Exemplo 1: **Auto Relacionamento -** Recupere o Nome e salário dos funcionários e o nome e salário de seus supervisores.

```
SELECT F.NOME AS NOMEFUNC, F.SALARIO AS SALARIOFUNC,
       S.NOME AS NOMESUP,  S.SALARIO AS SALARIOSUP
FROM FUNCIONARIO F LEFT OUTER JOIN FUNCIONARIO S ON
(F.CODFUNCSUPERVISOR = S.CODFUNC)
```

Exemplo 2: **Tabela Repetida** - Para cada departamento, recupere o nome do departamento, o nome do gerente e o total de funcionários que recebem mais de 500 reais lotados atualmente no departamento.

```
SELECT D.NOME, G.NOME AS GERENTE, COUNT(F.CODFUNC) AS TOTALFUNC
FROM DEPARTAMENTO D LEFT OUTER JOIN FUNCIONARIO G ON
(D.CODFUNCGERENTE = G.CODFUNC) LEFT OUTER JOIN LOTACAO L ON
(D.CODDEPARTAMENTO = L.CODDEPARTAMENTO) LEFT OUTER JOIN
FUNCIONARIO F ON (L.CODFUNC = F.CODFUNC)
WHERE L.DATAFIMLOTACAO IS NULL AND F.SALARIO > 500
GROUP BY D.NOME, D.CODDEPARTAMENTO, G.NOME
```

Exemplo 3: **Cláusula TOP** - Recupere o nome e salário dos 10 funcionários não demitidos com o maior salário na empresa.

```
SELECT TOP 10 NOME, SALARIO FROM FUNCIONARIO
WHERE DATADEMISSAO IS NULL ORDER BY SALARIO DESC
```

Exemplo 4: **Funções YEAR, MONTH, DAY** - Recupere o nome dos 20 produtos mais vendidos no ano atual.

```
SELECT TOP 20 P.NOME, SUM(I.QUANTIDADE) AS TOTALVENDIDOMES
```

FROM PRODUTO P INNER JOIN ITENSPEDIDO I ON (P.CODPRODUTO = I.CODPRODUTO) INNER JOIN PEDIDO PE ON (I.CODPEDIDO = PE.CODPEDIDO)
WHERE YEAR(PE.DATAPEDIDO) = YEAR(GETDATE())
GROUP BY P.NOME ORDER BY SUM(I.QUANTIDADE) DESC

Exemplo 5: **UNION/INTERSECT** - Recupere o código dos funcionários que são vendedores ou/e engenheiros (ou = UNION ; e = INTERSECT).
SELECT CODFUNC FROM VENDEDOR
UNION / INTERSECT
SELECT CODFUNC FROM ENGENHEIRO

Exemplo 6: **DISTINCT** - Recupere o código dos empregados (uma única vez) que trabalham nos projetos 1, 2 ou 3.
SELECT DISTINCT CODFUNC FROM TRABALHAEM WHERE CODPROJETO IN (1,2,3)

Exemplo 7: **Função Substring, convert, upper, ltrim, rtrim,**
SELECT SUBSTRING(NOME, 1, 10)
FROM FUNCIONARIO

SELECT SUBSTRING(CONVERT(CHAR(20), DATANASCIMENTO), 1, 10)
FROM FUNCIONARIO

SELECT UPPER(LTRIM(RTRIM(NOME)))
FROM FUNCIONARIO

Exemplo 8: **INSERT INTO** - Suponha que a tabela T_Exemplo abaixo já tenha sido criada:
INSERT INTO T_EXEMPLO (NOMEFUNC, NOMECARGO)
SELECT F.NOME, C.DESCRICAO FROM FUNCIONARIO F LEFT OUTER JOIN CARGO C ON (F.CODCARGO = C.CODCARGO)

Exercícios:

1)Recupere o nome e data de nascimento dos funcionários que nasceram no mês de junho.
2)Recupere o nome do projeto, o total de funcionários que não estão demitidos e já trabalharam no projeto e o nome do coordenador do projeto. Mostre todos os projetos não encerrados. A resposta deve estar ordenada pelo nome do projeto.
3)Recupere o nome e quantidade em estoque dos 3 produtos com o menor estoque atual.
4)Conte o número dos diferentes valores de salário contidos na tabela funcionário.
5)Usando a cláusula UNION, recupere o nome e salário dos funcionários que são gerentes de departamentos ou coordenadores de projetos.
6)Usando a cláusula INTERSECT, recupere o código dos funcionários que são supervisores e vendedores.
7)Reescreva a consulta anterior usando somente o operador JOIN. Lembre-se de usar o operador DISTINCT para retirar os dados duplicados.
8)Recupere o nome e salário dos funcionários não demitidos que tenham salário maior que o salário de seu supervisor.
9)Recupere o nome dos funcionários supervisores que tenham mais de 40 anos.
10)Recupere o nome e formação escolar dos vendedores que fizeram pedidos no ano de 2007.
11)Usando a função substring, recupere o nome e data de nascimento (sem exibir hora minutos e segundos) dos funcionários lotados no departamento 2.

2.3 CONSULTAS AVANÇADAS EM SQL: PARTE II

Exemplo 1: **IS NULL em Outer Joins** - Recupere o nome dos funcionários que não trabalham em nenhum projeto.

```
SELECT F.NOME
FROM FUNCIONARIO F LEFT OUTER JOIN TRABALHAEM T ON
        (F.CODFUNC = T.CODFUNC)
WHERE T.CODFUNC IS NULL
```

Exemplo 2: **IS NULL em Outer Joins** - Recupere o nome dos funcionários que não são vendedores, mas são coordenadores de projetos.

```
SELECT F.NOME
FROM FUNCIONARIO F INNER JOIN PROJETO P ON
(F.CODFUNC = P.CODFUNCCOORDENADOR)
LEFT OUTER JOIN VENDEDOR V ON (F.CODFUNC = V.CODFUNC)
WHERE V.CODFUNC IS NULL
```

Exemplo 3: Recupere o Nome e Salário dos funcionários que recebem o maior salário na empresa.

```
SELECT NOME, SALARIO FROM FUNCIONARIO
WHERE SALARIO = (SELECT MAX(SALARIO) FROM FUNCIONARIO)
```

Exemplo 4: Recupere o Nome e salário dos funcionários que ganham abaixo da média salarial dos gerentes de departamento. Mostre o resultado ordenado pelo nome do funcionário.

```
SELECT     NOME, SALARIO FROM FUNCIONARIO
WHERE      SALARIO <  (SELECT AVG(F.SALARIO)
                       FROM FUNCIONARIO F INNER JOIN DEPARTAMENTO D
                       ON F.CODFUNC = D.CODFUNCGERENTE)
ORDER BY NOME
```

Exemplo 5: (NOT IN) Recupere nome dos funcionários que não são gerentes:

```
SELECT NOME FROM FUNCIONARIO
WHERE CODFUNC NOT IN (SELECT CODFUNCGERENTE FROM DEPARTAMENTO
                      WHERE CODFUNCGERENTE IS NOT NULL)
```

Atenção: Algumas subconsultas podem ser reescritas sem uso de subconsultas.

Atenção: O resultado de uma subconsulta com o operador IN/NOT IN não pode retornar um valor NULL.

Exemplo 6: Recupere o nome dos funcionários que não são vendedores, mas são coordenadores de projetos.

```
SELECT NOME FROM FUNCIONARIO
WHERE CODFUNC NOT IN (SELECT CODFUNC FROM VENDEDOR)
AND CODFUNC IN (SELECT CODFUNCCOORDENADOR FROM PROJETO WHERE
                CODFUNCCOORDENADOR IS NOT NULL)
```

Exercícios:

1) Recupere o nome dos funcionários que não são coordenadores de projetos de duas formas distintas:
   a) Sem usar subconsulta, ou seja, usando somente LEFT OUTER JOIN.
   b) Com subconsulta, usando o operador NOT IN na clausula WHERE.
2) Recupere o nome do funcionário não demitido com o menor salário na empresa.
3) Recupere o nome dos gerentes de departamento que ganham acima da media salarial dos funcionários que são vendedores.
4) Recupere os nome dos dois produtos mais vendidos até hoje na empresa.
5) Recupere o nome do vendedor que fez o pedido de maior valor até o momento.
6) Recupere o nome dos supervisores que ganham acima da média salarial dos funcionários não demitidos.
7) Use o operador IN para formular uma subconsulta que recupere o nome e data de nascimento de cada um dos empregados que são coordenadores de projetos e tenham que tenham idade acima de 30 anos.
8) Refaça a consulta anterior usando somente operador INNER JOIN (Sem Subconsulta).
9) Recupere o nome dos funcionários que são supervisores e coordenadores de projeto, mas não são gerentes de departamentos.

2.4 CONSULTAS AVANÇADAS EM SQL: PARTE III

Exemplo 1: (EXISTS) Recupere o nome dos projetos que tenham pelo menos um funcionário trabalhando nele.

```
SELECT P.NOME FROM PROJETO P
WHERE EXISTS ( SELECT * FROM TRABALHAEM T
                    WHERE P.CODPROJETO = T.CODPROJETO)
```

Exemplo 2: (NOT EXISTS) Recupere o nome dos departamentos que não contenha funcionários lotados atualmente.

```
SELECT D.NOME FROM DEPARTAMENTO D
WHERE NOT EXISTS (SELECT * FROM LOTACAO L
                        WHERE D.CODDEPARTAMENTO = L.CODDEPARTAMENTO
                        AND L.DATAFIMLOTACAO IS NULL)
```

Exemplo 3: (EXCEPT) Recupere o nome dos vendedores que não são gerentes de departamentos.

```
SELECT NOME FROM FUNCIONARIO
WHERE CODFUNC IN (SELECT CODFUNC FROM VENDEDOR
                        EXCEPT
                        (SELECT CODFUNCGERENTE FROM DEPARTAMENTO))
```

Exemplo 4: (DELETE COM SUBCONSULTAS) Remova os funcionários que já foram demitidos.

```
DELETE FROM FUNCIONARIO
WHRE CODFUNC IN (SELECT CODFUNC FROM FUNCIONARIO WHERE
DATADEMISSAO IS NOT NULL)
```

Exemplo 5: (UPDATE COM SUBCONSULTAS) Aumente em 20% o salário dos funcionários que são vendedores

```
UPDATE FUNCIONOARIO SET SALARIO = SALARIO *1.2
WHERE CODFUNC IN (SELECT CODFUNC FROM VENDEDOR)
```

Exemplo 6: **Operador CASE**

```
SELECT NOME,
   CASE
     WHEN SALARIO < 500 THEN 'Salário menor que 500'
     WHEN SALARIO < 2000 THEN 'Salário maior ou igual a 500 e menor 2000'
     ELSE 'Salário maior ou igual a 2000'
   END
FROM FUNCIONARIO
```

Exemplo 7: (SUBCONSULTAS NA CLÁUSULA FROM) Recupere o nome dos projetos ainda não encerrados em que mais de três funcionários já trabalharam.

```
SELECT P.NOME FROM PROJETO P
INNER JOIN (SELECT CODPROJETO FROM TRABALHAEM
                GROUP BY CODPROJETO HAVING COUNT(CODFUNC) > 3) AS P3F
ON (P.CODPROJETO = P3F.CODPROJETO)
WHERE P.DATAFIM IS NULL
```

Exemplo 8: (SUBCONSULTAS NA CLÁUSULA SELECT) Recupere o nome dos funcionários e o total de projetos que cada um deles já trabalhou.

```
SELECT F.NOME, (SELECT COUNT(T.CODPROJETO) FROM TRABALHAEM T
                    WHERE F.CODFUNC = T.CODFUNC GROUP BY T.CODFUNC)
FROM FUNCIONARIO F
```

Exercícios:

1) Observe a seguinte consulta SQL:

```
SELECT NOME FROM FUNCIONARIO
WHERE DATADEMISSAO IS NULL AND
CODFUNC NOT IN (SELECT CODFUNCGERENTE FROM DEPARTAMENTO)
```

a) Explique o que esta consulta faz.
b) Reescreva a consulta usando o operador NOT EXISTS.
c) Reescreva a consulta de forma mais otimizada possível (sem uso de subconsulta).

2 O que a seguinte consulta faz?

```
SELECT F.NOME
FROM FUNCIONARIO F INNER JOIN DEPARTAMENTO D ON (F.CODFUNC = D.CODFUNCGERENTE)
WHERE NOT EXISTS (SELECT * FROM PROJETO P WHERE F.CODFUNC = P.CODFUNCCOORDENADOR) AND
F.SALARIO > 1000 AND F.CODFUNCSUPERVISOR IS NULL
```

a) Reescreva a consulta 1 usando-se o operador NOT IN.
b) Reescreva a consulta 1 sem uso de subconsulta. Porque ela é mais rápida que as anteriores?
c) Se no lugar de INNER JOIN fosse escrito LEFT OUTER JOIN o resultado final da consulta seria alterado? Justifique. Neste caso qual das duas consultas tenderia a ter melhor desempenho?

3)

a) Usando o operador EXCEPT recupere o nome dos vendedores que não são coordenadores de projetos.
b) Refaça a consulta anterior usando o operador NOT IN.
c) Refaça a consulta anterior usando o operador NOT EXISTS.
d) Refaça a consulta anterior sem usar subconsulta.
e) Qual das consultas anteriores é mais rápida? Justifique.

2.5 VIEWS

Views são tabelas derivadas de outras tabelas mas que não contêm linhas. Quando são executadas as consultas sobre as views os dados destas tabelas são preenchidos.

Exemplo 1:

```
CREATE VIEW VFuncCargoFormacao (NomeFunc, NomeCargo, NomeFormacao) AS
  SELECT F.Nome, C.Descricao, FE.Descricao
  FROM Funcionario F LEFT OUTER JOIN Cargo C ON (F.CodCargo = C.CodCargo)
          LEFT OUTER JOIN FormacaoEscolar FE ON (F.CodFormacao = FE.CodFormacao)

SELECT * FROM VfuncCargoFormacao
```

Vantagens das Visões:

1. **Mascaram a complexidade de consultas SQL**: As visões escondem a complexidade do design do banco de dados do usuário. Isso fornece aos desenvolvedores a capacidade de alterar o design sem afetar a interação do usuário com o banco de dados. Ou seja, o desenvolvimento e manutenção de aplicações ficam simplificados.
2. **Simplificam o gerenciamento de permissões do usuário**: Em vez de conceder permissão para os usuários consultarem colunas específicas em tabelas, os proprietários do banco de dados podem conceder permissão para usuários consultarem dados apenas através de visões.
3. **Aprimoram o desempenho**: As visões são pré-compiladas e permitem que se armazenem resultados de consultas complexas. Outras consultas podem usar esses resultados resumidos.

Exemplo 2:

```
CREATE VIEW VProjetoFuncionario
                    (CodProjeto, NomeProjeto, CodFunc, NomeFuncionario) AS
SELECT P.CodProjeto, P.Nome, F.CodFunc, F.Nome
FROM   Projeto P LEFT OUTER JOIN TrabalhaEm TE
          ON (P.CodProjeto = TE.CodProjeto)  RIGHT OUTER JOIN Funcionario F
          ON (TE.CodFunc = F.CodFunc)

/* Recupera a quantidade de funcionários por projeto */
SELECT NomeProjeto, Count(CodFunc) AS "Total de Funcionários"
FROM   VProjetoFuncionario
GROUP BY NomeProjeto

/* Recupera o nome dos Funcionários que não trabalham em nenhum projeto */
SELECT NomeFuncionario FROM VProjetoFuncionario
WHERE  CodProjeto IS NULL
```

Exemplo 3:

```
CREATE VIEW FolhaPgtoPorDepto (CodDepartamento, TotalFolha) AS
  SELECT D.CodDepartamento, SUM(F.Salario)
  FROM   Departamento D JOIN Lotacao L ON (D.CodDepartamento = L.CodDepartamento)
          JOIN Funcionario F ON (L.CodFunc = F.CodFunc)
  WHERE L.DataFimLotacao IS NULL
  GROUP BY D.CodDepartamento
```

Exemplo 4: DROP VIEW VFuncCargoFormacao

Exemplo 5:

```
SET DATEFORMAT dmy;

CREATE VIEW V_10_Funcionario_TotalPedidos_Ano_2007
-- Esta view recupere o 10 vendedores que mais venderam em 2007
AS
SELECT TOP 10 V.CodFunc, F.Nome, SUM(P.ValorTotalPedido) AS Total
FROM Vendedor V LEFT OUTER JOIN  Pedido P ON (V.CodFunc = P.CodFunc) JOIN
Funcionario F ON (V.CodFunc = F.CodFunc)
WHERE YEAR(DataPedido) = 2007
GROUP BY V.CodFunc, F.nome
ORDER BY Total DESC ;

SELECT * FROM V_10_Funcionario_TotalPedidos_Ano_2007
ORDER BY total ASC
```

Exemplo 6: Como as views podem melhorar a segurança de acesso ao Banco de Dados:

```
CREATE VIEW V_FuncSupervisor (NomeFunc, SalarioFunc, NomeSupervisor, SalarioSup)
AS
SELECT F.nome, F.Salario, S.nome, S.Salario
FROM Funcionario F LEFT OUTER JOIN Funcionario S ON (F.CodFuncSupervisor =
S.CodFunc);

DENY SELECT ON FUNCIONARIO TO BD;
GRANT SELECT ON V_FuncSupervisor TO BD;
```

Observações:

- O Projeto de quais Views um sistema deve ter é uma tarefa crítica em um grande sistema e deve ser feita pelo analista e DBA na fase do Projeto do sistema.

Exercícios:

1) a) Crie uma view de nome V1_SEU_RA que recupere o nome do funcionário e o total de projetos que cada funcionário trabalha.
b) Faça uma consulta na view anterior que recupere o nome dos funcionários que trabalham em mais de um projeto.

2) a) Crie uma view de nome V2_SEU_RA que recupere o nome dos funcionários que são supervisores e coordenadores de projeto, mas não são gerentes de departamentos.
b) Por que a view anterior não deveria ser criada num banco de dados real de uma empresa?

3) a) Crie uma view e nome V3_SEU_RA que recupere o código do funcionário não demitido, o nome do departamento de lotação atual e a data de nascimento do funcionário.
b) Faça uma consulta na view anterior que mostre o nome do departamento com até dois funcionários lotados com mais de 30 anos.
c) Usando a view do item a, proponha uma consulta SQL que recupere o nome do funcionário não demitido, o nome do departamento de lotação atual e o nome do cargo do funcionário.

2.6 STOREDS PROCEDURES

São funções armazenadas no BD, compiladas e disponibilizadas para execução pela aplicação. Podem receber parâmetros.

Exemplo 1:

```
CREATE PROCEDURE SPFolhaPagamento AS
BEGIN
  SELECT D.Nome, F.Nome, F.Salario
  FROM      Departamento D JOIN Lotacao L ON (D.CodDepartamento = L.CodDepartamento)
            JOIN Funcionario F ON (L.CodFunc = F.CodFunc)
  WHERE L.DataFimLotacao IS NULL
  ORDER BY D.Nome, F.Nome
END
```

Para executar a procedure depois dela criada: EXEC SPFolhaPagamento.

São criadas pelo DBA/Analista ou pelo próprio fabricando do SGBD para:

1. Encapsular regra de negócio e criar lógica do aplicativo reutilizável. Facilita a manutenção e alteração da aplicação, pois se a regra muda basta mudar na procedure.
2. Execução mais rápida para consultas repetitivas. Todas as instruções SQL tornam-se parte de um único plano de execução no servidor e uma vez executada, o plano de execução fica em memória cache.
3. Redução no tráfego de rede gerado pela aplicação. Não trafega código SQL.
4. Facilitar e centralizar o gerenciamento de permissões.
5. Esconder dos usuários os detalhes das tabelas, não sendo necessário o acesso às tabelas diretamente.
6. Eliminar a possibilidade de um hacker enviar instruções SQL incorporadas nos valores de parâmetro dos formulários da aplicação.

Exemplo 2:

```
CREATE PROCEDURE SPTotalFolhaDepartamento (@CodDepartamento INTEGER) AS
BEGIN
  SELECT D.Nome, SUM(F.Salario) AS "Total Folha"
  FROM  Departamento D JOIN Lotacao L ON (D.CodDepartamento = L.CodDepartamento)
     JOIN Funcionario F ON (L.CodFunc = F.CodFunc)
  WHERE D.CodDepartamento = @CodDepartamento AND L.DataFimLotacao IS NULL
  GROUP BY D.Nome
END
```

EXEC SPTotalFolhaDepartamento 1 ou EXEC SPTotalFolhaDepartamento 3

Exemplo 3:

```
CREATE PROCEDURE SPRemoveFuncionario (@CodFunc INTEGER) AS
BEGIN TRANSACTION
  DELETE FROM Dependente WHERE CodFunc = @CodFunc;
  DELETE FROM Lotacao WHERE CodFunc = @CodFunc;
  DELETE FROM TrabalhaEm WHERE CodFunc = @CodFunc;
  DELETE FROM Engenheiro WHERE CodFunc = @CodFunc;
  UPDATE Pedido SET CodFunc = NULL WHERE CodFunc = @CodFunc;
  DELETE FROM Vendedor WHERE CodFunc = @CodFunc;
  UPDATE Departamento SET CodFuncGerente = NULL WHERE CodFuncGerente = @CodFunc;
  UPDATE Projeto SET CodFuncCoordenador = NULL WHERE CodFuncCoordenador = @CodFunc;
  DELETE FROM Administrador WHERE CodFunc = @CodFunc;
  DELETE FROM Funcionario WHERE CodFunc = @CodFunc;
COMMIT
```

Exemplo 4: DROP PROCEDURE SPTotalFolhaDepartamento

Observações:

- O Projeto de quais SPs um sistema deve ter é uma tarefa crítica em um grande sistema e deve ser feita pelo analista e DBA na fase do Projeto do sistema.

Exercícios:

1) Como explicado em sala, uma Stored Procedure é útil para melhorar o desempenho de consultas repetitivas sobre um banco de dados.
a) Crie uma stored procedure, com o nome P_SeuRA_1 que receba como parâmetro o codigo do funcionário e retorne o nome do funcionário, o nome do departamento atual de lotacao e o nome seu supervisor, caso tenha.
b) Execute a procedure anterior para o funcionário de código 1 e 10.

2) a) Crie uma View com nome V_SeuRA_1 que exiba o codigo do Projeto, o nome do Projeto, o codigo do funcionário que trabalha no projeto e o total de horas que o funcionário trabalha no projeto.
b) Crie uma procedure com nome P_SeuRA-2 que receba como parametro o código do projeto e recupere o codigo do Projeto, o nome do Projeto, o codigo do funcionário que trabalha no projeto e o total de horas que o funcionário trabalha no projeto, para o projeto que foi passado como parametro.
c) Usando a View do item a, formule a sentença SQL que recupere o nome do projeto de código 1.
d) Usando a Procedure do item b, formule a sentença SQL que recupere o nome do projeto de código 1.
e) Explique qual estratégia é mais eficiente para execução da consulta que recupere o nome do projeto de código 1: o uso da view ou da procedure? Justifique.

3) Crie uma stored procedure, com o nome P_SeuRA_3 que receba como parâmetro o código do pedido e retorne o valor total do pedido e o nome do vendedor que tirou o pedido.

4) Crie uma stored procedure, com o nome P_SeuRA_6, que retorne o nome e a quantidade em estoque dos produtos. Mostre primeiro os produtos com a menor quantidade em estoque. Execute a procedure para testá-la.

5) a) Crie uma view de nome V_SeuRA que retorne o codigo do funcionário, o nome do funcionario, o nome do departamento de lotacao atual do funcionário e o nome de seu supervisor.
b) Faça uma consulta na view anterior para o funcionário de código 2.
c) Crie uma procedure de nome P_SeuRA que receba como parametro o código do funcionário e retorne o codigo do funcionário, o nome do funcionario, o nome do departamento de lotacao atual do funcionário e o nome de seu supervisor.
d) Execute a procedure anterior para o funcionário de código 2.
e) Crie uma Function de nome F_SeuRA que faça a mesma coisa que o item c. Execute a função para o funcionário de código 2.

6)a) Proponha uma view no banco de dados envolvendo no mínimo as tabelas Projeto e Pedido.
b) Explique quando esta view seria útil no banco de dados.

6) a) Proponha uma procedure no banco de dados que receba três parâmetros.
c) Explique quando esta procedure seria últil no banco de dados.

2.7 TRIGGER

São Storeds Procedures que são disparadas automaticamente pelo SGBD durante um evento de inserção, exclusão ou atualização de linhas nas tabelas do BD.
São úteis para:

- Controlar restrições de domínio que depende de valores dinâmicos.
- Realizar tarefas em cascata antes ou após um INSERT, DELETE ou UPDATE (Exemplos: atualização do estoque ou implementação de opções de Cascade Delete, Set NULL, etc).
- Atualizar atributos derivados de tabelas (Ex: Total de dependentes de um Funcionário).
- Auditar ações realizadas pelos usuários.

Nota: Em SQL Server quando um trigger é disparado são criadas as tabelas DELETED e/ou INSERTED que contém as linhas removidas e inseridas. Por exemplo, se o trigger for disparado em função de um UPDATE a tabela DELETED conterá as linhas antes da atualização e a tabela INSERTED conterá as linhas depois da atualização.

Exemplo 1: Trigger para implementar uma restrição de domínio (SQL Server)

```
CREATE TRIGGER TGVerificaSalarioMinimo ON Funcionario FOR INSERT, UPDATE AS
BEGIN
IF (SELECT Salario FROM INSERTED) < (SELECT SalarioMinimo FROM Parametros)
   BEGIN
     PRINT 'Erro! Salário do funcionário é menor que o salário mínimo '
     ROLLBACK TRANSACTION
   END
END
```

Exemplo 2: Trigger para implementar uma atualização em cascata (SQL Server):aumenta o salário de um funcionário que se tornou gerente:

```
CREATE TRIGGER TGAtualizaSalarioGerenteNovo ON Departamento FOR INSERT AS
BEGIN
  UPDATE Funcionario
  SET Salario = Salario + INSERTED.GratificacaoGerente
  FROM INSERTED
  WHERE Funcionario.CodFunc = INSERTED.CodFuncGerente
END
```

Exemplo 3: Trigger que reduz o salário de um gerente cujo departamento que gerenciava foi removido:

```
CREATE TRIGGER TGAtualizaSalarioGerenteDepartamentoRemovido ON Departamento
FOR DELETE AS
BEGIN
  UPDATE Funcionario
  SET Salario = Salario - DELETED.GratificacaoGerente
  FROM DELETED
  WHERE Funcionario.CodFunc = DELETED.CodFuncGerente
END
```

Exemplo 4: Trigger que aumenta o salário do funcionário que se tornou o novo gerente e reduz o salário do funcionário que deixou de ser gerente:

```
CREATE TRIGGER TGAtualizaSalarioGerenteRemovido ON Departamento FOR UPDATE AS
BEGIN
  IF UPDATE("CodFuncGerente")
   BEGIN
       /* Aumenta o salário do funcionário que se tornou o novo gerente */
       UPDATE Funcionario
         SET Salario = Salario + INSERTED.GratificacaoGerente
         FROM INSERTED
         WHERE Funcionario.CodFunc = INSERTED.CodFuncGerente
       /* Reduz o salário do funcionário que deixou de ser gerente */
       UPDATE Funcionario
         SET Salario = Salario - DELETED.GratificacaoGerente
         FROM DELETED
         WHERE Funcionario.CodFunc = DELETED.CodFuncGerente
   END
END
```

Exemplo 5: Atualização de atributos derivados: Cria a Trigger que calcula o atributo derivado totaldependentes da tabela Funcionário.

```
CREATE TRIGGER TGAtualizaTotalDependentes
ON Dependente FOR INSERT, DELETE AS
BEGIN
  IF (SELECT COUNT(*) FROM INSERTED) = 1 /* Houve uma inserção */
     BEGIN
       UPDATE Funcionario
       SET TotalDependentes = TotalDependentes + 1
       WHERE CodFunc = (SELECT CodFunc FROM INSERTED)
     END
  ELSE
     IF (SELECT COUNT(*) FROM DELETED) = 1 /* Houve uma remoção */
     BEGIN
       UPDATE Funcionario
       SET TotalDependentes = TotalDependentes - 1
       WHERE CodFunc = (SELECT CodFunc FROM DELETED)
     END
END
```

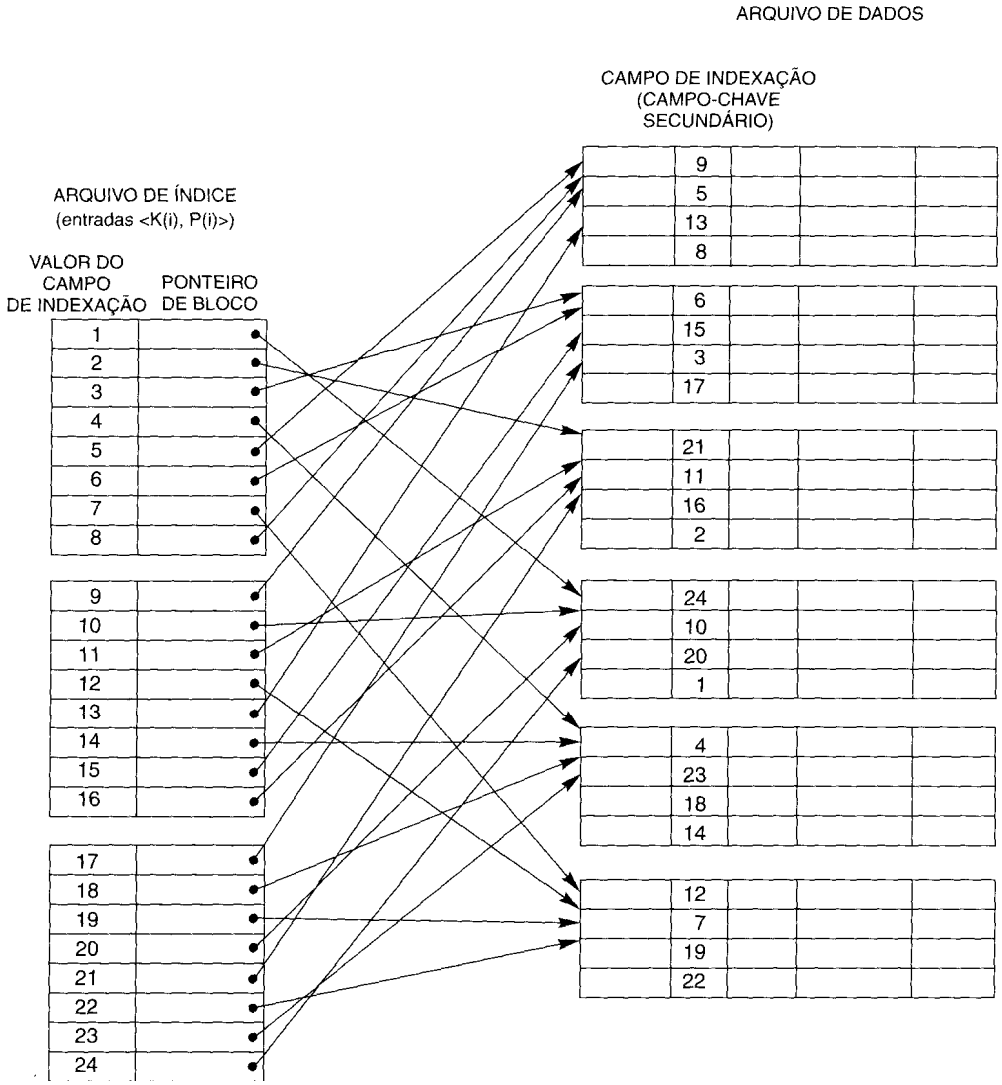
Exemplo 6: DROP TRIGGER TGAuditaSalario

- O Projeto de quais Triggers um sistema deve ter é uma tarefa crítica em um grande sistema e deve ser feita pelo analista e DBA na fase do Projeto do sistema.
- Um DBA conhecer a sintaxe de criação de triggers do fabricante do SGBD para usar todo o potencial desta característica.
- Cuidado: Uma trigger pode disparar outra trigger em cascata.

3 Tópicos Avançados em BD

3.1 OTIMIZAÇÃO (ÍNDICES)

- Índice é uma tabela que implementa uma estrutura de dado, como árvore B, através de uma ou mais colunas especificadas no índice, com o objetivo de evitar o acesso seqüencial.



- Tipos de índices:
  - ♦ Índices implícitos: são os índices que são automaticamente criados pelo SGBD quando o objeto é criado. Ex: os atributos chaves (PKs).
    Ex: CREATE INDEX I_Funcionario ON Funcionario (CodFunc)
  - ♦ Índices simples: são aqueles baseados em uma única coluna da tabela.
    Ex: CREATE INDEX I_FuncCargo ON Funcionario (CodCargo)
    Ex: CREATE INDEX I_ProjDataInicio ON Projeto (DataInicioProjeto)
  - ♦ Índices compostos: são aqueles baseados em duas ou mais colunas de uma tabela.
    Ex: CREATE INDEX I_DeptoFunc ON Lotacao (CodDepto, CodFunc)
    Ex: CREATE INDEX I_FuncCargoSalario ON Funcionario (CodCargo, Salario)
  - ♦ Índices únicos: implementam valores únicos em uma coluna semelhante a uma PK.
    Ex: CREATE UNIQUE INDEX I_CPF ON Funcionario (CPF)

Removendo índices: DROP INDEX Projeto.I_Descricao

Critérios gerais para otimização de um Banco de Dados:
1) Crie índices em:
- Colunas utilizadas como chaves estrangeiras. Exceções: A tabela não deve conter poucas linhas ou os valores da chave estrangeira não deve conter poucos valores distintos.
- Colunas freqüentemente utilizadas para unir (JOIN) tabelas: Foreign Keys.
- Colunas freqüentemente utilizadas como condições em uma consulta, desde que a consulta não retorne mais de 80 % das linhas nas tabelas.

2) Crie Views e/ou Storeds Procedures sobre as consultas mais freqüentes do BD.

Quando você deve evitar criar índices:
1) Em colunas que contenham poucos valores únicos, como sexo e status.
2) Em colunas de consultas que recuperem uma alta percentagem das linhas das tabelas.
3) Um índice geralmente ocupa muito espaço em disco podendo ser maior que a própria tabela. Portanto veja se espaço de armazenamento é um problema no seu sistema.
4) Se uma coluna sofrer constantes modificações índices nesta coluna devem ser avaliados sobre pena de o algoritmo de indexação sobrecarregar o sistema.
5) Se existem processos de carga em tabelas deve-se primeiro remover os índices desta tabela e depois criá-los novamente, de preferência quando o SGBD estiver ocioso.

Nota: O Projeto de quais Índices um sistema deve ter é uma tarefa crítica em um grande sistema e deve ser feita pelo analista e DBA na fase do Projeto do sistema.

3.2 SEGURANÇA EM BANCO DE DADOS

- Um banco de dados possui usuários que são autenticados usando logins específicos ou logins validados pelo sistema operacional.
- Para estabelecer uma conexão com SGBD é necessário ter um login no Servidor de BD.
- Um Database pode ter usuários que podem ser colocados em um grupo (role).
- Todo SGBD possui um grupo default chamado public.

Exemplos:
```
USE master
CREATE LOGIN BD WITH PASSWORD = 'sql' , DEFAULT_DATABASE = Pessoal,
DEFAULT_LANGUAGE = brazilian
USE Pessoal
CREATE USER BD FOR LOGIN BD
CREATE ROLE ALUNOS
SP_ADDROLEMEMBER "ALUNOS", "BD"
SP_DROPROLEMEMBER "ALUNOS", "BD"
SP_DROPROLE "ALUNOS"
```

- Existem dois tipos de permissões que controlam as ações de um usuário:

A) Permissão de declaração:
CREATE DATABASE, CREATE TABLE, CREATE RULE, CREATE VIEW, CREATE PROCEDURE e CREATE INDEX.

Exemplos:
1) GRANT CREATE TABLE, CREATE VIEW TO cesar
3) GRANT CREATE PROCEDURE TO public

B) Permissão de objetos:
SELECT, INSERT, UPDATE, DELETE e EXECUTE (Storeds Procedures)

Exemplos:
1) GRANT UPDATE ON projeto TO cesar, iremar
2) GRANT SELECT ON funcionario (CodFunc, Nome, Endereco, Telefone) TO public
4) GRANT DELETE ON dependente TO marcia WITH GRANT OPTION
5) GRANT EXECUTE ON SPFolhaPagamento TO iremar

C) Opção REVOKE (remove uma ou mais permissões já concedidas)
Exemplos:
1) REVOKE CREATE TABLE FROM cesar
3) REVOKE SELECT ON funcionario FROM cesar

Esquemas (Scheme) em banco de Dados (Microsoft SQL Server)

1. Um esquema (schema) é um container para objetos do banco de dados (ANSI SQL-92).
2. Facilita o gerenciamento de permissões.
3. Quando um objeto é criado se não for definido o esquema a que ele pertence, automaticamente o objeto vai pertencer ao esquema dbo.

Exemplos:
a) Cria um esquema de nome RH no banco de dados Pessoal.
USE PESSOAL;
CREATE SCHEMA RH
Atenção: Esquemas são criados em bancos de dados.

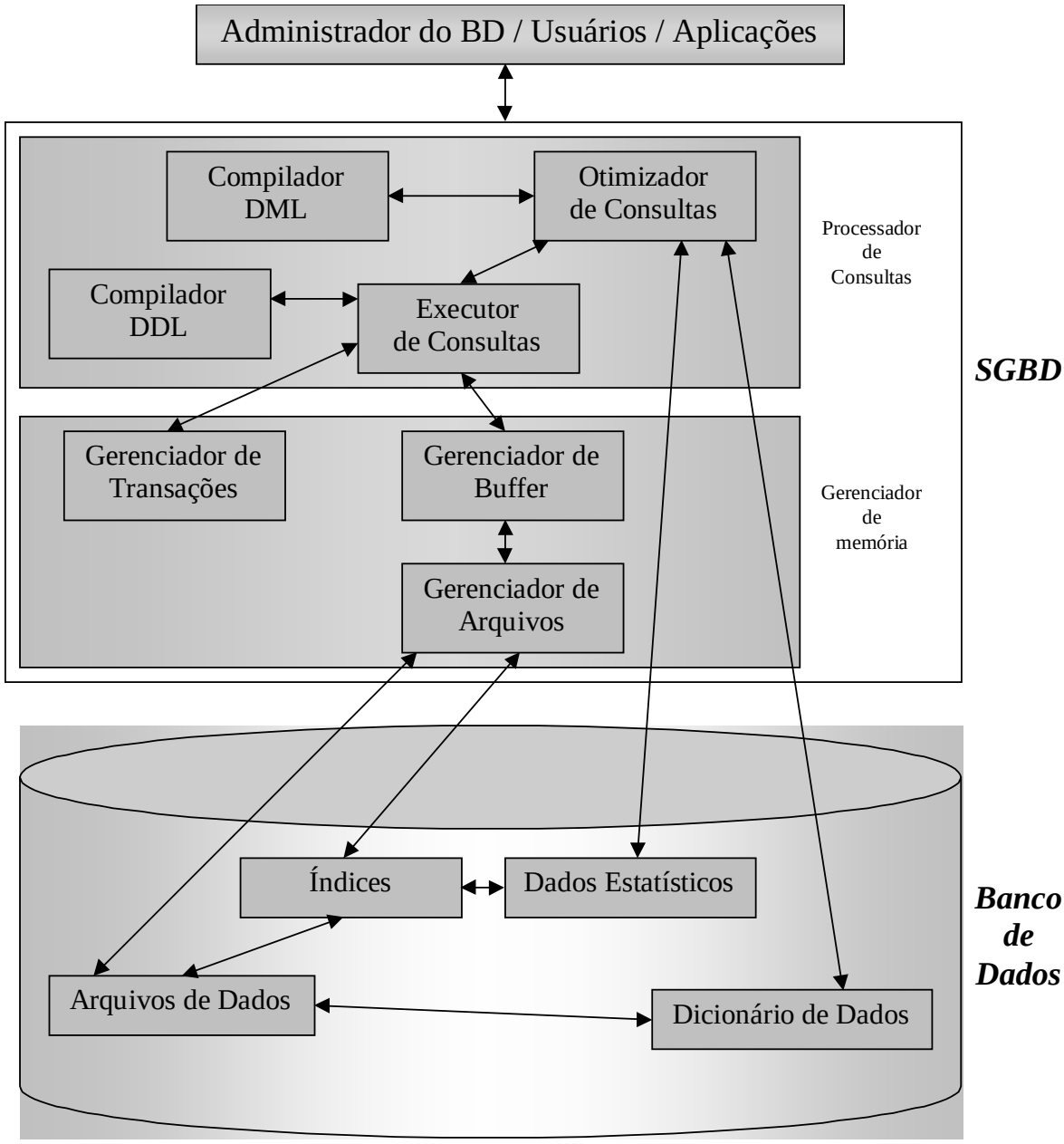
b) Transfere o esquema da tabela Cargo para o esquema RH
ALTER SCHEMA RH TRANSFER dbo.cargo;
SELECT * FROM RH.cargo

c) Cria uma tabela no esquema RH:
USE pessoal;
CREATE TABLE RH.Pedido (CodPedido int, Valor money, DataPedido datetime)

d) Concede permissões em esquemas:
GRANT SELECT ON SCHEMA::RH TO guest
DENY DELETE ON SCHEMA::RH TO guest

3.3 ARQUITETURA DE SGBDS

A figura a seguir ilustra, de forma simplificada, os componentes típicos de um SGBD:



Adaptação de "*Sistema de Banco de Dados*", Henry F. Korth et al.

Estes módulos são descritos a seguir:

- ***Compilador DML***: Módulo responsável pela tradução de comandos DML (linguagem de manipulação de dados do tipo SQL), que podem ter sido embutidos em um programa de aplicação ou enviados pelo Administrador do Banco de Dados.
- ***Compilador DDL***: Módulo que traduz comandos DDL (linguagem de definição de dados do tipo SQL), que podem ser submetidos pelo DBA ou usuários avançados do BD.
- ***Otimizador de consultas***: Módulo que tem a tarefa de produzir um plano de execução que permita escolher uma estratégia eficiente para executar a consulta submetida ao SGBD.
- ***Executor de consultas***: Módulo que gera e executa o código de baixo nível correspondente à consulta submetida ao SGBD.

- ***Gerenciador de transações***: Módulo responsável pelo controle de concorrência, segurança, controle de restrições e recuperação de falhas das transações submetidas ao SGBD.
- ***Gerenciador de buffer***: Módulo responsável por manusear buffers na memória principal mantendo ali os dados mais acessados pelo SGBD.
- ***Gerenciador de arquivos***: Módulo responsável por transferir dados de baixo nível (arquivos físicos) entre o disco e o armazenamento central do computador (memória principal).

Diferença entre Administrador de banco de dados e Administrador de Dados

- ***Administrador do banco de dados*** : Também chamado de DBA - Database Administrator, representa o superusuário do SGBD, responsável por autorizar o acesso ao banco de dados, por coordenar e monitorar sua utilização e por adquirir recursos de hardware e software quando necessário. Ajuda na solução de problemas como violação da segurança ou por tempos demorados de resposta do sistema.
- ***Administrador de dados***: É o profissional especializado nos dados da organização visando o desenvolvimento integrado dos sistemas de informação da empresa. Não precisa conhecer necessariamente o funcionamento do SGBD mas deve ser um profissional especializado na regra de negócios da organização ajudando o analista na definição do modelo conceitual.

Exercícios: Para cada questão abaixo escolha a opção correta apresentando a justificativa da sua escolha.

1) Em relação à arquitetura de um SGBD incorreto afirmar que:

a) O **gerenciador de buffer** interage como gerenciador de arquivos.

b) O **otimizador de consulta** interage diretamente com o dicionário de dados do SGBD.

c) O **gerenciador de transações** não resolve problemas de concorrência de acesso aos dados do Banco de Dados.

d) O **executor de consultas** realiza a execução de uma sentença SQL submetida ao SGBD.

2) Todas as afirmações abaixo são incorretas exceto:

a) O **dicionário de dados** é um dos componentes de um SGBD.

b) O **gerenciador de buffer** procura manter em memória principal os dados mais acessados pelas pelos usuários de um SGBD.

c) O **otimizador de consulta** não garante uma boa estratégia na execução de uma consulta SQL.

d) Uma das tarefas centrais de um **administrador de dados** é configurar um SGBD após sua instalação.

3) São funções de um DBA exceto:

a) Definir a estrutura de armazenamento do banco de dados e a estratégia de acesso aos dados.

b) Implementar os controles de segurança e integridade no SGBD.

c) Monitorar o SGBD.

d) Propor o DER.

4) São funções de um Analista de Sistemas exceto:

a) Montar a lista de eventos que um sistema deve responder.

b) Definir as restrições de domínio do banco de dados.

c) Definir um plano de contingência para recuperação do banco de dados.

d) Definir as entidades que um sistema deve ter.

4. Exercícios de Revisão

Exercícios de revisão 1:

1) a) Explique o que a seguinte consulta recupera:

SELECT P.NOME

FROM PRODUTO P WHERE NOT EXISTS (SELECT * FROM ITENSPEDIDO I WHERE P.CODPRODUTO = I.CODPRODUTO)

b) Reescreva a consulta anterior usando o operador NOT IN.

c) Qual das duas consultas anteriores é mais otimizada? Justifique.

d) Reecreva a consulta do item a usando somente o operador LEFT OUTER JOIN (Sem o uso de subconsulta).

e) Por que a consulta proposta no item d tende a ser mais otimizada?

2) Proponha o codigo SQL mais otimizado possível que recupere o nome dos produtos que ainda não foram vendidos neste ano (use o operador YEAR).

3) a) Proponha uma view de nome V_SeuRA que recupere o nome do funcionario, o codigo do funcioinario e o nome de seu cargo. Mostre todos os funcionarios mesmo que nao tenha cargo.

b) Usando a view anterior proponha uma nova consulta SQL que use a view anterior e que recupera o nome do funcionario, o nome de seu cargo e o nome de sua formacao escolar.

4) a) Proponha uma procedure de nome P_SeuRA que recupere o nome do funcionario e o nome de seu supervisor. Mostre todos os funcionarios.

c) Crie outra procedure de nome P_SeuRA_2 que receba o codigo do funcionario como parametro e retorne o nome do funcionario e o nome de seu supervisor.

5) Quando devemos usar triggers num banco de dados?

6) Proponha o codigo SQL que recupere o nome dos funcionarios que receba acima da media salarial dos funcionarios que sao vendedores.

Exercícios de revisão 2:

Atenção: Responda as perguntas SQL de forma mais otimizada possível.

1) Recupere o nome dos gerentes de departamento que ganham acima da média salarial dos funcionários coordenadores de projeto.

2) Proponha uma view que recupere o nome do departamento e a média salarial dos funcionários lotados atualmente no departamento.

3) Proponha uma procedure que receba como parâmetro a código do vendedor e retorne o nome do vendedor e a soma total dos valores dos pedidos retirados pelo vendedor no mês atual.

4) Crie um índice sobre a coluna data do pedido na tabela pedido.

5) Monte uma consulta SQL em que o SGBD poderia usar o índice anterior.

6) Crie um trigger que atualize o atributo TotalDependentes da tabela Funcionário, caso seja removido um dependente na tabela Dependente.

7) Proponha a consulta SQL que recupere o nome dos funcionários gerentes de departamentos, e se possuir, o nome de seu supervisor.

8) Proponha o comando SQL que dê permissão para o usuário BD remover linhas na tabela Projeto.

9) Proponha o comando que SQL dê permissão do usuário BD de consultar a todas as colunas da tabela Funcionario com exceção da coluna salário.

10) Crie um índice único na coluna CodFuncGerente da tabela Departamento. Explique em que situação este índice poderia ser útil.

Exercícios de revisão 3:

1) O que a seguinte consulta faz?

SELECT F.NOME

FROM FUNCIONARIO F INNER JOIN DEPARTAMENTO D ON (F.CODFUNC = D.CODFUNCGERENTE)

WHERE NOT EXISTS (SELECT * FROM PROJETO P WHERE F.CODFUNC = P.CODFUNCCOORDENADOR) AND F.SALARIO > 1000 AND F.CODFUNCSUPERVISOR IS NULL

2) Reescreva a consulta 1 usando-se o operador NOT IN.

3) Reescreva a consulta 1 sem uso de subconsulta. Porque ela é mais rápida que as anteriores?

4) Quais Índices poderiam ser criados na consulta 1 de forma a torná-la mais rápida? Crie-os.

5) Os Índices criados na consulta 4 são úteis para a consulta 3?

6) Se no lugar de INNER JOIN fosse escrito LEFT OUTER JOIN o resultado final da consulta seria alterado? Justifique. Neste caso qual das duas consultas tenderia a ter melhor desempenho?

7) Quais os “grants” deveriam ser executadas para o usuário José poder executar a consulta 1 ?

8) Crie uma procedure com parâmetro CODFUNC para a consulta 1.

9) Crie uma VIEW para a consulta 1.

10) Faça uma consulta na VIEW anterior que recupere os funcionários gerentes que não são coordenadores de projetos, mas recebem acima de 1000 e que não tenham supervisores e nem sejam vendedores.

11) Aponte uma vantagem e uma desvantagem de uma trigger.

12) Qual o principal objetivo de um esquema em BD?

Exercícios de revisão 4:

1) EXPLIQUE QUANDO UM INDICE É USADO PELO SGBD NUMA CONSULTA

2) O QUE É UMA TRIGGER DDL? PARA QUE SERVE?

3) O QUE FAZ O MODLO GERENCIADOR DE BUFFER DE UM SGBD

4) QUAIS OS 3 PASSOS QUEUMA SENTENÇA SQL PASSA QUANDO E SUBMETIDA EM UM SGD

5) É POSSIEL RETIRAR A PERMISSAO DE LEURA NUMA COLUNA DE UMA TABELA PARA O USUARIO DO BD? JUSIFIQUE

6) QUAL A DIFERENÇA ENTRE LOGIM E USER NO SQL SEVER

7) QUAL A DIFERENÇA ENTRE DBA E AD.

8) O QUE SIGNIFICA DIZER QUE UMA PROCEDURE É PRE COPILADA

9) O QUE FAZ O MÓDULO GERENCIADOR DE TRANSAÇÕES DE UM SGBD?